

CLAIM AMENDMENTS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A method, comprising:

determining a new enqueue slot of a circular queue having N slots into which a queue element may be enqueued;

setting a last enqueue slot ("LES") pointer currently designating an old enqueue slot to designate the new enqueue slot after determining the new enqueue slot;

determining whether the circular queue is full via executing a check comparing relative positions of the new enqueue slot and a current dequeue slot ("CDS"), wherein the check determines whether enqueueing the queue element into the new enqueue slot would result in an overflow condition of the circular queue; [[and]]

dropping the queue element, if the overflow condition would result from enqueueing the queue element into the new enqueue slot; and

enqueueing the queue element into the new enqueue slot, if the circular queue is not full.

2. (Cancelled)

3. (Currently Amended) The method of claim [[2]] 1, further comprising:

setting a last enqueue slot ("LES") pointer currently designating an old enqueue slot to designate the new enqueue slot after determining the new enqueue slot;

~~dropping the enqueue element, if the overflow condition would result from enqueueing the queue element into the new enqueue slot; and~~
resetting the LES pointer to designate the old enqueue slot, if the overflow condition would result from enqueueing the queue element.

4. (Original) The method of claim 3, wherein executing the check further comprises determining whether the following relation is true:

$$((CDS^N - LES^N) \bmod N) < M,$$

wherein CDS^N represents $CDS \bmod N$, LES^N represents $LES \bmod N$, and M represents a number less than N .

5. (Original) The method of claim 4 wherein M is equal to or greater than a maximum number of slots that may be enqueued with queue elements during a delay period for updating a dequeue counter.

6. (Original) The method of claim 1 wherein determining the new enqueue slot of the circular queue into which the queue element may be enqueued comprises determining the new enqueue slot according to a pre-sort deficit round robin enqueueing scheme.

7. (Currently Amended) The method of claim [[2]] 1 wherein each of the N slots of the circular queue [[can]] buffers multiple queue elements corresponding to multiple logical queues, wherein the queue element corresponds to a particular one of the multiple logical

queues, and wherein the new enqueue slot corresponds to the particular one of the multiple logical queues.

8. (Original) The method of claim 7 wherein determining whether the circular queue is full comprises determining whether enqueueing the queue element into the new enqueue slot of the circular queue would result in an overflow condition of the particular one of the multiple logical queues.

9. (Currently Amended) A method, comprising:

dequeuing a queue element from a current dequeue slot ("CDS") of a circular queue having N slots;

designating a new CDS;

determining whether the circular queue is empty via executing a first check comparing relative positions of the new CDS and a last enqueued slot ("LES"), wherein executing the first check includes determining whether the following relation is true:

$((CDS^N - LES^N) \bmod N) < M$, wherein CDS^N represents $CDS \bmod N$, LES^N represents $LES \bmod N$, and M represents a number less than N; and

setting the LES to the new CDS, if the circular queue is determined to be empty.

10. (Original) The method of claim 9 wherein the CDS is designated by a CDS pointer, wherein designating the new CDS comprises incrementing the CDS pointer to designate the new CDS, wherein the LES is designated by a LES pointer, and wherein

setting the LES to the new CDS comprises setting the LES pointer to designate the new CDS, if the circular queue is determined to be empty.

11. (Original) The method of claim 9 wherein determining whether the circular queue is empty further comprises executing a second check prior to executing the first check, the second check comprising:

determining whether an enqueue count is equal to a dequeue count.

12. (Original) The method of claim 11, further comprising incrementing the dequeue count after dequeuing the queue element from the CDS of the circular queue.

13. (Cancelled)

14. (Original) The method of claim 9 wherein M is equal to or greater than a maximum number of slots of the circular queue that may be enqueued with queue elements during a delay period for updating a dequeue counter.

15. (Currently Amended) The method of claim 9 wherein each of the N slots of the circular queue [[can]] buffers multiple queue elements corresponding to multiple logical queues, wherein the queue element corresponds to a particular one of the multiple logical queues, and wherein LES corresponds to the particular one of the multiple logical queues.

16. (Original) The method of claim 15 wherein determining whether the circular queue is empty via executing the first check comprises determining whether the particular one of the multiple logical queues is empty via executing the first check.

17. (Currently Amended) A ~~machine-accessible~~ computer-accessible storage medium that provides instructions that, if executed by a machine, will cause the machine to perform operations comprising:

dequeueing a first queue element from a current dequeue slot ("CDS") of a circular queue having N slots, the CDS designated by a CDS pointer;

incrementing the CDS pointer to designate a new CDS; and

determining whether the circular queue is empty after the incrementing via executing a first check comparing relative positions within the circular queue designated by the CDS pointer and a last enqueued slot ("LES" pointer"), wherein executing the first check includes determining whether the following relation is true:

$((CDS^N - LES^N) \bmod N) < M$, wherein CDS^N represents $CDS \bmod N$, LES^N represents $LES \bmod N$, and M represents a number less than N.

18. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 17, further providing instructions that, if executed by the machine, will cause the machine to perform further operations, comprising:

setting the LES pointer currently designating an old enqueue slot of the circular queue to designate a new enqueue slot of the circular queue into which a second queue element may be enqueued;

determining whether enqueueing the second queue element into the new enqueue slot would result in an overflow condition of the circular queue via re-executing the first check after setting the LES pointer to designate the new enqueue slot.

19. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 18, further providing instructions that, if executed by the machine, will cause the machine to perform further operations, comprising:

enqueueing the second queue element into the new enqueue slot, if the overflow condition would not result from enqueueing the second queue element into the new enqueue slot;

dropping the second enqueue element, if the overflow condition would result from enqueueing the second queue element into the new enqueue slot; and

resetting the LES pointer to designate the old enqueue slot, if the overflow condition would result from enqueueing the second queue element into the new enqueue slot.

20. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 18, further providing instructions that, if executed by the machine, will cause the machine to perform a further operation, comprising determining the new enqueue slot of the circular queue into which the second queue element may be enqueuee.

21. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 20, wherein determining the new enqueue slot of the circular queue into

which the second queue element may be enqueued comprises determining the new enqueue slot according to a per-sort deficit round robin queuing scheme.

22. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 17, further providing instructions that, if executed by the machine, will cause the machine to perform further operations, comprising:

setting the LES pointer to designate the current dequeue slot, if the circular queue is determined to be empty.

23. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 22 wherein the determining whether the circular queue is empty further comprises executing a second check prior to executing the first check, the second check comprising:

determining whether an enqueue count is equal to a dequeue count, wherein the enqueue count is incremented each time a queue element is enqueued and the dequeue count is incremented each time a queue element is dequeued.

24. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 18 wherein executing and re-executing the first check comparing the relative positions within the circular queue designated by the CDS pointer and the LES pointer comprises determining whether the following relation is true:

$$((CDS^N - LES^N) \bmod N) < M$$

wherein CDS^N represents CDS mod N, LES^N represents LES mod N, and M represents a number less than N.

25. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 18 wherein each of the N slots of the circular queue $[[can]]$ buffers multiple queue elements corresponding to multiple logical queues, and wherein the first queue element, the second queue element, and the LES pointer correspond to a particular one of the multiple logical queues.

26. (Currently Amended) The ~~machine-accessible~~ computer-accessible storage medium of claim 25 wherein determining whether the circular queue is empty comprises determining whether the particular one of the logical queues is empty and wherein determining whether enqueueing the second queue element would result in an overflow condition of the circular queue comprises determining whether enqueueing the second queue element would result in an overflow condition of the particular one of the logical queues.

27. – 35. (Cancelled)